Technical Report 1548
Revision 1
January 1994

Environment/Tool Integrator for Software Development

Version 1.2

N. T. Tran
R. H. Mumm

EXECUTIVE SUMMARY

OBJECTIVE

The objective of this research was to develop a windowing framework for accessing the diverse collections of software tools needed by software development projects. A major research goal was to develop an environment/tool integrator (ETI) that can easily be customized to satisfy the unique needs of specific projects. The purpose of this report is to (1) explain why the ETI was developed, (2) describe it, and (3) provide instructions for its installation and use.

RESULTS

This report describes the second released version of the ETI, version 1.2. The new capabilities provided by this version include X/Motif front-ends and context-sensitive help for many tools; conformance to the most recent NRaD user interface specifications; and porting to additional UNIX-based platforms, besides Sun Workstations. Version 1.2 was submitted to the Software Technology for Adaptable Reliable Systems (STARS) Asset Source for Adaptable Reliable Software (ASSET) library in the Spring of 1994.

RECOMMENDATIONS

The most significant recommendations are:

1. More help is needed from human factors experts during development of user interfaces for DoD software and CASE tools. The operation of these user interfaces can—and must be—simplified.
2. Users should aid in cleaning up Ada repositories. Tools that do not work, or are of marginal quality, should be removed.
3. The Ada Joint Program Office, National Aeronautics and Space Administration, or other government organization(s) should provide support to AdaNet to improve the quality of tools in the repository. Improvements that would benefit the entire Ada community include X/Motif front-ends, modifications to increase portability, and other enhancements.

CONTENTS

FIGURES

1.0 INTRODUCTION

This document describes the environment/tool integrator (ETI), version 1.2, developed under the Software Engineering Environment Prototypes (SWEEP) task of the Software Engineering for Command Control and Communications (C^3) Systems project. The ETI is a windowing framework for accessing software tools that can be easily customized to satisfy the needs of a project. The ETI allows users to define (1) the appropriate tool categories for a specific project and (2) the interfaces for accessing individual tools. Tools that may be accessed include computer-aided software engineering (CASE) tools, software-engineering environments, reuse libraries, project-specific tools, public domain tools, and utility tools. (For a list of the acronyms and abbreviations used in the report, refer to Chapter 8.)

The ETI provides a convenient way of displaying the names of all tools available to a project development team. It supports multiple programming languages as well as different applications (e.g., C^3 tactical and corporate information-management [CIM] applications).

The ETI executes on the following Unix-based computers: SPARC 1 (OS: SunOS 4.1.3), Silicon Graphics (OS: IRIX 4.05H), DEC Alpha (OS: OSF 1.3), IBM 6000 (OS: A1X2.3), and SPARC 10 (OS: SunOS 5.2). Tools may reside on various workstations in a local area network (LAN).

The ETI, version 1.2, was submitted to the Software Technology for Adaptable Reliable Systems (STARS) Asset Source for Software Engineering Technology (ASSET) library. It is available from the STARS ASSET library at no cost and with unlimited distribution. Ada and C public-domain tools available with version 1.2 include an Ada pretty printer, Ada line counter, Ada body stubber, C pretty printer, C line counter, and a C function prototype generator. Utility tools include editors, a calculator, clipboard, mailer, and others.

The ETI was developed in-house by the Naval Command, Control and Ocean Surveillance Center Research, Development, Test and Evaluation Division (NRaD). Version 1.1 was used at software development sites for the Operations Support System (OSS) project at NRaD.

Major enhancements to the ETI, version 1.2, include the following:

1. Porting to many Unix-based computers. (Version 1.1 executed only on Sun workstations.)
2. X/Motif front-ends and context-sensitive help for three Ada tools (Ada Primitive Compilation Order Tool (APRICOT), PAGER, and line counter).
3. Conformance to the most recent version of the Navy Command and Control user interface specifications (reference 2). This includes a capability to invoke tools from the keyboard, the use of default buttons, and the use of different colors for editable text and other widgets (user interface components).
4. A C-function prototype generator for generating header files for C source code.

This report includes the following information for version 1.2 of the ETI:

- Background information
- Characteristics
- User instructions
- Lessons learned
- Recommendations
- Descriptions of the tools included

Potential users of the ETI and this report are software developers who use Unix-based computers.

2.0 BACKGROUND

The need for the ETI was recognized by observing the ways in which software was being developed for several projects at NRaD and at other Navy laboratories. The following problems were identified that can be solved or minimized by using the ETI:

1. Project software engineers were sometimes unaware of potentially useful tools available for software development, even when they resided on project computers. Very useful tools that could have been used were not being applied.
2. Most commercially available environments have extensibility limitations. Examples of such limitations include:
    - Users cannot add their own tools, because the environment is predetermined and cannot be modified by the user.
    - Tool integration is possible, but difficult and time consuming.
    - Tool integration is possible only by accessing the newly integrated tool through layers of menus.
    - Upon the release of a new version of an environment, users must reintegrate project tools.
3. Commercial environments are frequently tied to a computer language or a to vendor's compiler. NRaD and Navy projects often use multiple programming languages.
4. Licensing and cost considerations preclude the widespread use of commercial environments.

The ETI solves each of these four problems by:

1. Providing a way that project users can view and access all available project tools. This is especially helpful for programmers who are new to a project.
2. Providing an alternative to integrating tools into commercial environments. Tools may be integrated into the ETI and accessed from it. Multiple commercial environments and reuse libraries can also be accessed from it.
3. Not tying into any one language or compiler. Multiple compilation systems may be accessed.
4. Making it public-domain software. The ETI is available from the STARS ASSET library.

3.0 CHARACTERISTICS

This section describes the key features of the ETI, and also lists the required hardware and support software.

3.1 KEY FEATURES

The ETI:

- Enables the use of software tools, environments, and reuse libraries together, in an integrated manner, for software development.
- Offers easy customization by users. Project members can define tool categories and tool interfaces by creating a setup table. Knowledge of Graphical User Interface (GUI) builder tools and X/Motif is not required.
- Includes enhanced public domain Ada and C software development tools and utility tools.
- Provides interfaces to commercial environments and tools.
- Conforms to Navy Command and Control user interface specifications (reference 1).
- Includes an on-line help facility.
- Offers portability across platforms (Unix).

3.2 REQUIRED HARDWARE AND SOFTWARE

The hardware and software required by the ETI are as follows:

- The ETI executes on SPARC 1 (OS: SunOS 4.1.3), Silicon Graphics (OS: IRIX 4.05H), DEC Alpha (OS: OSF 1.3), IBM 6000 (OS: A1X2.3), and SPARC 10 (OS: SunOS 5.2).
- The ETI and included tools require approximately 16 megabytes of disk space. They will execute with the minimum memory provided by any of the above systems. The executables run on SPARC 1 (OS: SunOS 4.1.3). To produce executables for the other systems (mentioned in the previous paragraph), the source code must be recompiled.
- The X/Motif (X11R5 libraries and Motif 1.2) is required.

4.0 LESSONS LEARNED

During development of the ETI, the authors learned several lessons, which are covered below.

4.1 VALUE OF DEMONSTRATIONS

Task members found a transition project and users for the ETI by rapidly developing a functional prototype that was demonstrated repeatedly. Task members felt the initial prototype offered enough functionality for project developers to recognize its potential. Demonstrations were given to key personnel of candidate projects at NRaD and other Navy laboratories. Users in private industry were found by giving demonstrations at national Armed Forces Communications and Electronics Association (AFCEA) conferences.

These demonstrations have led to the ETI improvements resulting from user feedback. Examples of helpful suggestions include (1) making the entire top-level window tailorable and (2) providing a context-sensitive help capability. The final capability allows a user to get help on a particular field by positioning the cursor on the field and then by clicking the mouse button.

4.2 ADVANTAGE OF FINDING USERS EARLY

Early in the development of the ETI, the NRaD Operations Support System (OSS) project members made many valuable suggestions, which included providing the capability for subcategories of tools; and reducing the size of the top-level window, so that other application windows were visible.

4.3 MOTIF HELP FROM USENET BULLETIN BOARD

USENET (Users' Network) is an internationally distributed bulletin board supported mostly by Unix machines. We found it to be an excellent source for detailed technical information about Motif. Posting messages with questions on USENET was more successful than using other sources. For more information on how to read and post news, type: `man rn` and `man Pnews`, respectively, on Unix machines that have access to those programs.

4.4 GUI BUILDERS

GUI builders are interactive tools for building graphical user interfaces. At the time actual development started on the ETI, a number of GUI builders were considered for use. The only GUI builder that then supported Ada was TAE+, and it lacked maturity. Builder Xcessory was chosen, because we had used it previously, it provided the capabilities needed, and it was reasonably priced. In the past several years, TAE+ has improved substantially, and quality GUI builders have become available that support Ada.

4.5 ENHANCEMENT TO USER INTERFACE SPECIFICATIONS

We were able to conform to the NRaD user interface specifications (reference 1) by using Motif—except for one case. The specifications say that the background color of the scrolled list widget is light gray and the background color of the slider in the scroll bar is light blue. We found this was not possible using the OSF/Motif list widget and recommend that the standards be changed. No other difficulties were found with other aspects we used.

4.6 TAILORING

Three tailoring approaches were considered. In the first approach, users would employ a GUI builder to customize the ETI to meet their needs. The second was to provide an automated method for constructing the windowing framework from a setup table. With the third, the second approach would be extended by building the setup table from an interactive program. The second approach was followed. The first approach was not chosen, because it requires the user to know Motif and how to use a GUI builder. And the third was not done because of time constraints and because it would probably not be significantly easier for users.

4.7 SOURCES FOR PUBLIC DOMAIN TOOLS

The source of most Ada public domain tools was the AdaNet repository in Morgantown, WV. Sources for C tools were found in the Archie bulletin board (reference 2). The utility programs are from the X11R5 libraries. Considerable time and effort was spent compiling and executing tools from the Ada repositories. Some tools worked, some did not work well, and some did not work at all. The quality of tools and components in the repositories must be upgraded with the help of the user community. Upgrading needs to include the removal or enhancement of those units that do not compile nor execute properly. Furthermore, when a tool or component is vastly inferior to one with equivalent or better functionality, the inferior unit should be removed.

The AdaNet repository contained six Ada pretty printers. Only pretty printer 4 and pretty printer 6 were written in valid Ada. Pretty printer 6 was selected because task members could not get pretty printer 4 to execute properly. Note that pretty printer 6 still contains some minor bugs.

Two Ada body stubbers were found in AdaNet. Neither of them execute using a validated Ada compiler; however, body Stubber 2 was modified by task members so it would run on validated Ada compilers. Several Ada line counters were examined in the two repositories, with the line counter, `File_Checker`, selected. Several bugs were found in it and fixed.

The C line counter selected, kdsi, was located in the Archie bulletin board (reference 2). One minor bug (for an uninitialized variable) was fixed before integrating the line counter into the ETI. Archie can be accessed by rlogin_ing or telnet-ing to archie.ans.net (147.225.1.2) or archie.sura.net (128.167.254.179) with username `archie` and no password.

The C pretty printer currently used is from SunOS 4.1.3.

Appendix D contains information needed for contacting or connecting to these repositories.

4.8 HUMAN FACTORS INVOLVEMENT

A human factors expert, Dr. K. Fernandes, periodically evaluated the ETI.
She provided invaluable comments about simplifying the user interface. Her user interface standard was followed (reference 1). Her comments related to consistency, readability, and other aspects covered in the interface standard.

5.0 RECOMMENDATIONS

5.1 RECOMMENDATIONS FOR FUTURE WORK

The following list gives our recommendations for future ETI research and development.

1. Make the following improvements to the ETI—by adding:
   - A tool-search capability to find the category in which the tool is located.
   - Comprehensive error messages for the setup table.
   - A context-sensitive help capability.
2. Include additional Ada tools. Those being considered for inclusion are Halstead metrics, McCabe metrics, a path analyzer, performance analyzer, and a statement profiler from AdaNet. Others that have wide application are also candidates.
3. Include C++ tools and write Motif front-ends for these tools. They include a pretty printer, line counter, and others that are in the public domain and have wide application.
4. Include additional C tools and write Motif front-ends for them. These tools include a cross referencer—and others.
5. Write Motif front-ends for commercial CASE tools to facilitate their use. These front-ends facilitate using startup scripts to define tool parameters. Tools may include CADRE Teamwork and Interactive Development Environments' (IDE) Software Thru Pictures.
6. Upgrade the ETI to conform to revised standards.

5.2 GENERAL RECOMMENDATIONS

These general recommendations resulted from our observations while developing the ETI:

- Developers of DoD software and CASE tool developers should have human factors specialists evaluate early prototypes of their user interfaces; such an evaluation can help to eliminate awkwardness and inconsistencies, and simplify interfacial use.
- Software developers must use standards to help them produce user interfaces consistent in appearance and operation. We suggest using the NRaD user interface specifications (reference 1) that we used.

6.0 REFERENCES

1. Fernandes, K. 1992. “User Interface Specifications for Command and Control Systems,” ver. 1.2. Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA.

2. Deutsch, P., A. Emtage, B. Heelen, M. Parker. 1992. “Archie Database,” Archie Group, McGill University, Montreal, Canada.

7.0 ANNOTATED BIBLIOGRAPHY

MountainNet 1992. “AdaNET Service, ver. 3.0, ASV3, User’s Guide for ASCII Terminals,” Morgantown, WV.

The report contains instructions for accessing and extracting components from the AdaNET repository. The AdaNET repository is sponsored by the National Aeronautics and Space Administration Technology Utilization Division through the Johnson Space Center. It is maintained by the AdaNET staff at MountainNET.

MountainNet. 1993. “ASV3 Catalog of Holdings—Software Components,” Morgantown, WV.

This report is a list of the components contained in the AdaNET repository. An abstract is included for each component.

Open Software Foundation. 1993. “OSF/Motif Style Guide,” rev. 1.2 (for OSF/Motif rel. 1.2). Prentice Hall, Englewood Cliffs, NJ.

The style guide contains a framework of behavior specifications to help developers build products conforming to the OSF/Motif user interface.

Quercia V., and T. O’Reilly. 1993. “X Window System User’s Guide,” vol. 3, O’Reilly and Associates, Inc., Sebastapol, CA.

This guide describes the X window system concepts, use of various client applications, and customization of the X environment.

STARS. 1993. “ASSET Users Guide,” Morgantown, WV.

This users guide, available from within the ASSET library access software, contains instructions for accessing the library. The ASSET library is maintained by IBM and SAIC under the STARS program.

Tran, N., and H. Mumm. 1992. “Environment/Tool Integrator for Software Development,” ver. 1.1. NRaD TR 1548 (Aug). Naval Command, Control and Ocean Surveillance Center RDT&E Division, San Diego, CA.

This report describes the ETI, ver. 1.1, provides user instructions, and discusses lessons learned during the ETI development.

8.0 ACRONYMS AND ABBREVIATIONS

ABOM	—	Ada Bit-Oriented Message Handler Project
ALS/N	—	Ada Language System/Navy
AFCEA	—	Armed Forces Communications and Electronics Association
APRICOT	—	Ada Primitive Compilation Order Tool
ARG	—	APRICOT Report Generator
ASG	—	APRICOT Script Generator
ASSET	—	Asset Source for Software Engineering Technology
CARDS	—	Central Archive for Reusable Defense Software
CASE	—	Computer-Aided Software Engineering
CIM	—	Corporate Information Management
CRSS	—	C^3I Reusable Software System
C^3	—	Command, Control and Communications
DoD	—	Department of Defense
DTC-2	—	Desk Top Computer-2
ETI	—	Environment/Tool Integrator
GUI	—	Graphics User Interface
IDE	—	Interactive Development Environments
LAN	—	Local area network
MACA	—	Management Assistant Corporation of America
MIT	—	Massachusetts Institute of Technology
NASA	—	National Aeronautics and Space Administration
NED	—	Navy Command and Control System Ashore Editor
NRaD	—	Naval Command, Control and Ocean Surveillance Center Research, Development, Test and Evaluation Division
OSS	—	Operations Support System
QIC	—	Quarter-Inch Cartridge
SAIC	—	Science Applications International Corporation
SAINT	—	Shared Adaptive Internetworking project
STARS	—	Software Technology for Adaptable Reliable Systems
SWEEP	—	Software Engineering Environment Prototypes
TAC-3	—	Tactical Advanced Computer 3
TI	—	Texas Instruments
X	—	The X Window System

APPENDIX A
USING THE ENVIRONMENT/TOOL INTEGRATOR (ETI)

A.1 INSTALLATION

This appendix gives instructions for installing the ETI either from a quarter-inch cartridge (QIC) tape provided by NRaD or when extracting files from the STARS ASSET library, which is funded by the STARS program. Products in the library are available to software practitioners in industry, government, and academia.

These instructions assume the user has a basic understanding of the Unix operating system commands and is running the Unix C-shell or Bourne shell for the SPARCstation 1, using SunOS 4.1.3. To run on other platforms, the ETI must be recompiled before applying the following installation instructions. The recompilation step also applies to the C and Ada tools that come with the ETI. The Ada Compilation Order Maker will only compile with a Verdix compiler.

Before following the steps below, choose the directory in which the ETI is to be installed by using the Unix command, `cd`. Then follow the appropriate set of instructions.

A.1.1 Installation from a QIC Tape Provided by NRaD

1. Unload Files from Tape

   Insert the ETI tape into the tape drive. Unload the files on the tape using the command below. This will automatically create the directory, `sweep1.2`.

   ```
   tar xvf /dev/rst0 ./sweep1.2.tar
   ```

2. Set Up the Environment Variables

The environment variables are set up in the file, `.cshrc`, for C-shell; or, `.profile`, for Bourne shell. This file must be in the user's home directory. Note in the instructions below, the notation <*path*> indicates that the installer must specify the appropriate complete path name. Place the following commands at the bottom of this file, using an editor:

For C-shell:

```
setenv SWEEP_HOME <path>/sweep1.2
set path=($SWEEP_HOME/bin $path)
setenv XAPPLRESDIR $SWEEP_HOME/app-defaults/
```

For Bourne shell:

```
SWEEP_HOME=<path>/sweep1.2
PATH=$SWEEP_HOME/bin:$PATH
XAPPLRESDIR=$SWEEP_HOME/app-defaults
export SWEEP_HOME XAPPLRESDIR PATH
```

Next, type the following to set the environment variables:

For C-shell:

```
source ~/.cshrc
```

For Bourne shell:

```
sh ~/.profile
```

Confirm proper installation by invoking the ETI and noting that the top-level window is displayed on the user's screen. The invocation step and a picture of the top-level window are given in paragraph A.2.1.

A.1.2 Installation from the STARS ASSET Library

1. Extract and Transfer Files to the Host Computer.

   a. Get an Account on the STARS ASSET Computer.

      Before installing the ETI from the STARS ASSET library, obtain an account on the ASSET computer. To do this, call (304) 594–1762 for a user form. Fill it out and return it to the ASSET library system administrator. The ASSET library mail address is provided in Appendix D.

   b. Retrieve the Environment/Tool Integrator.

      First, connect your host computer to the ASSET computer with the command

      `telnet source.asset.com` (192.131.125.10)

      and log on using your userid and password.

      The ASSET main menu will be displayed automatically on the screen, once you log on the system. Select the `ASSET Reuse Library` option from the main menu to access the library. To search for the ETI, select `ASSET search, ASSET specific data`, and `Name` options from menus on the screen; then enter `Environment/Tool Integrator`. Press the "S" key to activate the search. Finally, press the "E" key to extract the ETI files from the library and copy them to your current directory on the ASSET computer.

      If you need detailed instructions for using the ASSET library, go back to the main menu, select "User's Guide," and browse to the appropriate instructions.

   c. Transfer the Environment/Tool Integrator Files.

      Transfer the files to your host computer, using ftp, Kermit, or other file transfer software from the ASSET computer. Documentation containing detailed information for performing file transfers is widely available.

2. Uncompress and Restore File.

   Enter the following commands:

   ```
   uncompress sweep1.2.tar.Z
   tar xvf sweep1.2.tar
   ```

   The first command uncompresses the single ETI file, `sweep1.2.tar.Z`, and replaces it with the file, `sweep1.2.tar`. The second creates the individual files and automatically creates the directory, `sweep1.2.`

3. Set Up the Environment Variables.

   Use the same steps given in paragraph A.1.1., subparagraph 2.

A.2 EXECUTION

A.2.1 Invoking the SWEEP Environment/Tool Integrator

Start the ETI (in any directory) by typing

```
sweep
```

(The ETI was developed under the Software Engineering Environment Prototypes [SWEEP] task of the Software Engineering for Command, Control, and Communications Systems project.)

The initial top-level window, shown in figure A-1, will appear on your monitor. This window may be tailored to suit the specific needs of individual projects. Tailoring, discussed in section A.3, may change details of the window's appearance.

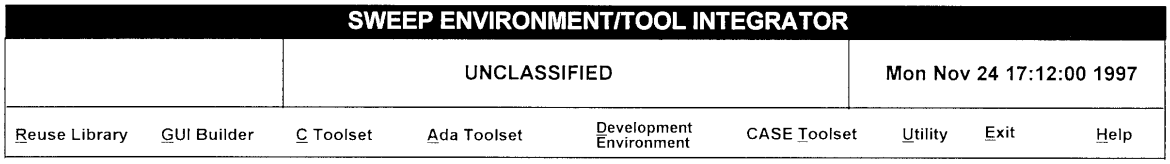

Figure A-1. Top-level window.

A.2.2 Top-Level Window Description

At the top of figure A-1, the line below the title line, `SWEEP ENVIRONMENT/TOOL INTEGRATOR`, contains the security classification position, which, in this example, is `UNCLASSIFIED`. This position represents the highest level of classification for any project data. The position to its right shows the current system date and time.

The next line shows the seven tool categories, as well as the `Exit` and `Help` capabilities.

Figure A-2 shows the individual tools within each tool category and the options available under `Exit` and `Help`. In this figure, all options are displayed. (This is done for explanatory purposes only. In actual use, tools are displayed for only one category at a time.)

In figure A-2, the tools displayed in bold print (black on the monitor) are included with version 1.2 of the ETI. Those not included appear in italic print in the figure (gray on the monitor). Tools displayed in gray have not been installed. Most tools displayed in gray are commercially available and, if needed by the project, must be purchased. Note that figure A-2 shows that the current (ETI, version 1.2) C Toolset contains a function prototype generator, line counter, and pretty printer; the Ada Toolset contains a body stubber, compilation order maker, line counter, pager, and pretty printer; and the utility tools comprise a calculator, clipboard, command shell, editors, E-mailer, and file manager. The right arrow next to `Editor` indicates that multiple editors are provided. The underline characters in the tool and tool category names indicate that the user can use the keyboard to invoke the tools. Keyboard invocation is discussed in paragraph A.2.3.

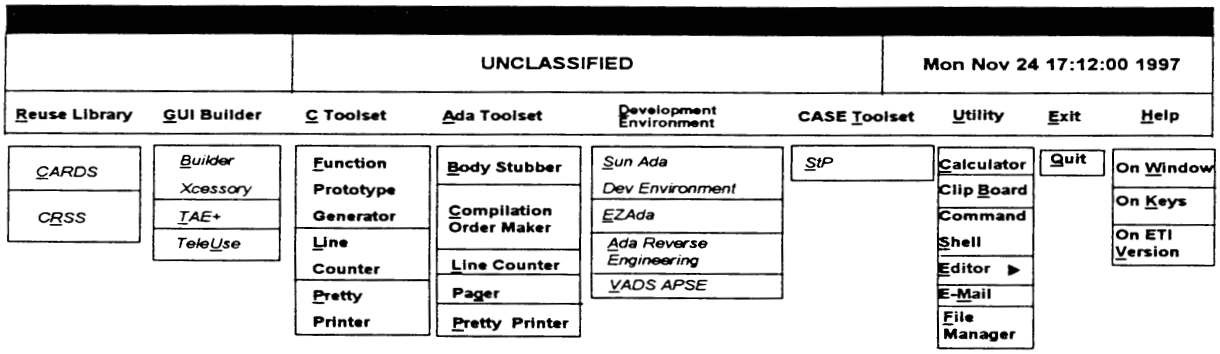

Figure A-2. Top-level window with all tools and options displayed.

Next, note the options that are provided under `Help`, on the right-hand side of the window. The `Help` capability provides the user with (1) help instructions for the top-level window, (2) help instructions for using mnemonics, and (3) ETI version information.

The ETI is initially configured to include the seven tool categories and the tools indicated in figure A-2. This configuration may be changed by following the tailoring instructions given in section A.3. The number of tool categories, names of the tool categories, and tools within each may be changed. All projects probably will want to keep the `Utility` and `Help` category. `Exit` is the only category the user cannot customize.

Figure A-3 shows the subcategory capability. The right arrow pointing from `Editor` indicates that multiple editors may be selected. The editors within this subcategory are the NED, Emacs, and Xedit editors. A brief description of the NED is contained in Appendix B. A more complete description is contained in reference 1, Appendix A. NED complies to the Navy Command and Control user interface specifications (reference 2, Appendix A). Emacs is the GNU Emacs editor that is in the public domain. Xedit is an X editor also in the public domain.

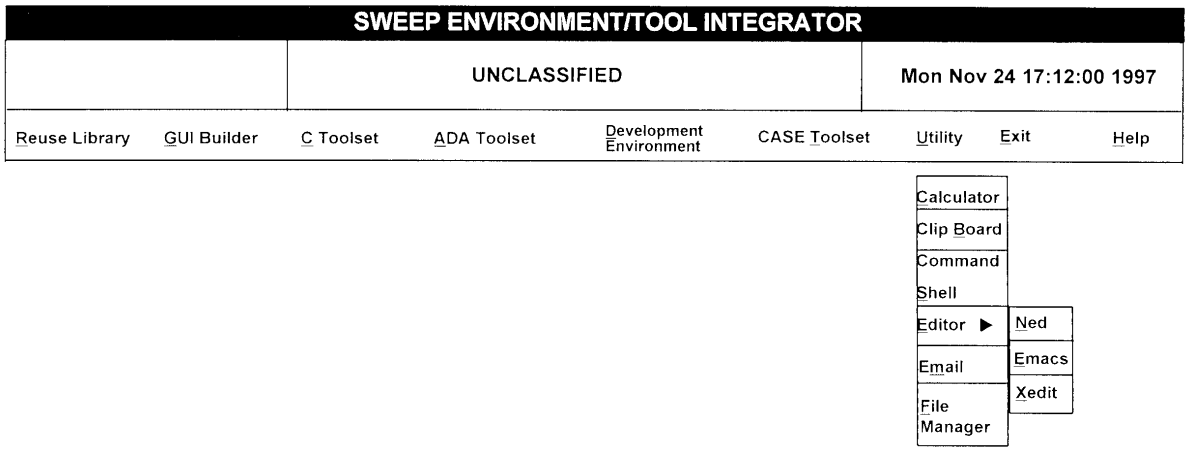

Figure A-3. Top-level window with subcategory of editors displayed.

A.2.3 Tool Invocation

To invoke an individual tool, click the left mouse button on the tool category and then click the left mouse button on the appropriate tool. Tools can also be invoked with the keyboard by

holding down the Meta key (usually located next to the space bar) and pressing the key whose letter is underlined in a tool category. The tool menu under the tool category displays on the screen. Invoke a tool from the menu by using the underlined character in the tool name. For example, to invoke the `Line Counter` under the `C toolset` category, hold down the Meta key and press "C" to display the tool menu. Then press "L" to invoke the Line Counter. (Appendix B contains a description of ETI tools.)

A.3 TAILORING

Users are allowed to define the name of the top-level window, the classification of project data, the number and names of tool categories, and the tools within each. However, to comply with the Navy Command and Control user interface specifications (reference 1), the ETI should not have more than 10 categories, plus `Help`, and should not have more than 10 tools within each category.

A.3.1 Version 1.2 as Delivered

The window and tool categories shown in figure A-2 are automatically produced from the setup table shown in figure A-4. The setup table, contained in the file, `sweep_config`, was automatically loaded during installation. This table is free format (i.e., at least one blank space separates each field).

A.3.1.1 Setup Table. At the top of the setup table (figure A-4), the field, `SWEEP ENVIRONMENT/TOOL INTEGRATOR`, provides the name of the top-level window, which is displayed at the top of figure A-2. The next field in figure A-4, `UNCLASSIFIED`, gives the highest level of classification of the project data.

Next, note the tool category names and the tool names on the left side of the table. Both must be enclosed in quotes. Next to them are the mnemonics that provide a shortcut to invoke tools using the keyboard. A mnemonic of a tool or a tool category can be any single character, usually the first character, of a tool or a tool category name. The mnemonic must be enclosed in quotes. No tool categories can use the same mnemonic. Each tool in a tool category must use a mnemonic that is unique within the tool category, and each tool in a subcategory must use a mnemonic that is unique within the subcategory. If the user chooses not to have a mnemonic, a pair of contiguous quotes must be present. The mnemonics are underlined when displayed on the screen.

To the right of each tool's mnemonic is the executable field, which can be represented in three different ways.

1. It can be the path name with the executable file name. Enclose such a string in quotes.

   For example, look about two thirds of the way down figure A-4. In the `Utility` category, for the `Editor` subcategory, for Ned, the path name to the NED executable is the path name including the executable file, `$SWEEP_HOME/bin/ned.`

2. It can be the executable file name enclosed in quotes (without the path name). In this case, the path name must be set in the `.cshrc file.`

```
"SWEEP ENVIRONMENT/TOOL INTEGRATOR"

"UNCLASSIFIED"

"Reuse Library"  "R":
       "CARDS"  "C"                          "",
       "CRSS"     "R"                        "";
"GUI Builder"  "G":
       "Builder Xcessory"  "B"               "",
       "TAE+" "T"                            "",
       "TeleUse"  "U"                        "";
"C Toolset"  "C":
       "Function Prototype Generator"  "F" "c_fn_prototype",
       "Line Counter"  "L"                   "c_line_counter",
       "Pretty Printer"  "P"                 "c_pretty_printer";
"ADA Toolset"  "A":
       "Body Stubber"  "B"                   "stubber.exe ",
       "Compilation Order Maker"  "C"        "com",
       "Line Counter"  "L"                   "ada_line_counter",
       "Pager"  "g"                          "pager",
       "Pretty Printer"  "P"                 "pretty_print.exe";
"Development Environment"  "D":
       "Sun Ada Dev Environment"  "S"        "",
       "EZ-Ada"  "E"                         "",
       "Ada Reverse Engineering"  "A"        "",
       "VADS APSE"  "V"                      "";
"CASE Toolset"  "T":
       "StP"  "S"                            "";
"Utility"  "U":
       "Calculator"  "C"                     "xcalc -geometry +10+120",
       "Clip Board"  "B"                     "xclipboard geom +10+120",
       "Command Shell"  "S"                  "xterm -geom +10+120",
       "Editor" "E":
            "Ned"  "N"                       "$SWEEP_HOME/bin/ned",
            "Emacs"  "E"                     "xterm -fn 10x20 -e emacs",
            "Xedit"  "X"                     "xedit" ;  .,
       "E-mail"  "m"                         "xmh -geo +10x120",
       "File Manager"  "F"                   "file_manager";
Help:
       "On Window"  "W"                      "help.window",
       "On Keys"  "K"                        "help.keys",
       "On ETI Version"  "V"                 "help.version";
```

Figure A-4. Example of setup table.

For example, in the `C Toolset` category (toward the top of figure A-4), for `Line Counter`, the executable field is `c_line_counter`, which indicates that the path name is in `.cshrc.`

3. It can be a pair of contiguous quotes representing a null field, which indicates the executable file name has not been specified. This may be used, for example, to represent a place holder for the tool, indicating it has not been purchased or it is not presently available. The `CARDS` and `CRSS` library executable fields show null fields.

Subcategories of tools can be defined. For example, the `Utility` category has the `Editor` subcategory that consists of the three editors: Ned, Emacs, and Xedit.

Also note that `Exit` category—the only noncustomizable category—does not appear in the setup table.

Finally, note the file names, `help.window, help.keys` and `help.version`, on the bottom right of the table under the `Help` category. These files contain text for the on-line help and are automatically put in the directory `$SWEEP_HOME/help` during installation. If you want to add more help options, place all their text files in the directory `$SWEEP_HOME/help.`

When creating a setup table:

- Enclose each tool category name in quotes, followed by a colon (except for the `help` category).
- Enclose each tool name and mnemonic in quotes.
- Separate tool information with the delimiter comma.
- End category information with a semicolon.
- When the name of the executable file is not given within quotes, note that the tool will appear in gray on the monitor.
- When there is a subcategory of tools, the rules are analogous to those for a category of tools.
- Enclose the subcategory name in quotes, followed by a mnemonic and a colon.
- Enclose each tool name in quotes.
- Separate tool information with the delimiter comma, and include the end of the subcategory with a semicolon.

A.3.1.2 X Instructions. In figure A-4, some executable file names are followed by additional instructions. These are command-line X instructions the user may or may not need.

A short explanation of these instructions is provided here. For a detailed explanation, refer to the on-line users manual, `man pages` (reference 3, Appendix A), which is included with the X Window System from the Massachusetts Institute of Technology (MIT).

Find the executable file `xcal` about two thirds of the way down figure A-4, on the right. The expression `–geometry +10+120` is an X instruction that tells xcal that the starting x,y pixel position for xcal is 10,120. This is needed to ensure that the `xcal` windows do not overlay the ETI top-level window (figure A-1). You may arbitrarily select any x pixel position that lies on the screen. The y pixel position is critical for positioning the tool's windows below the ETI top-level window.

Now, further down in figure A-4, find the `Emacs` editor. The expression `xterm –fn 10x20 -e emacs` indicates (1) the font size for the emacs editor is a 10-by-20 pixel fixed-width font (the emacs font size was increased from the default size to increase readability) and (2) emacs is executed inside the xterm window.

A.3.2 An Example of Tailoring to a Specific Project

Figure A-5 shows the top-level window for a hypothetical project (when all tools and options are displayed). This project does not use Ada, but uses C and FORTRAN. Note how this

top-level window differs from the one given in figure A-2. The label at the top of the screen is now `PROJECT X.` The category, `FORTRAN Toolset,` was created (the fourth category). For Project X, a number of project-specific tools are used, so the next tool category, `Project X Tools,` was created. Finally, two different CASE tools are used, Code Center and Cadre Teamwork. Figure A-6 shows the changes that were made to the setup table to tailor the ETI to a specific project. The changes were made to the file, `sweep_config`. After making the appropriate changes, type:

```
sweep
```

to bring up the ETI, as shown in figure A-5.

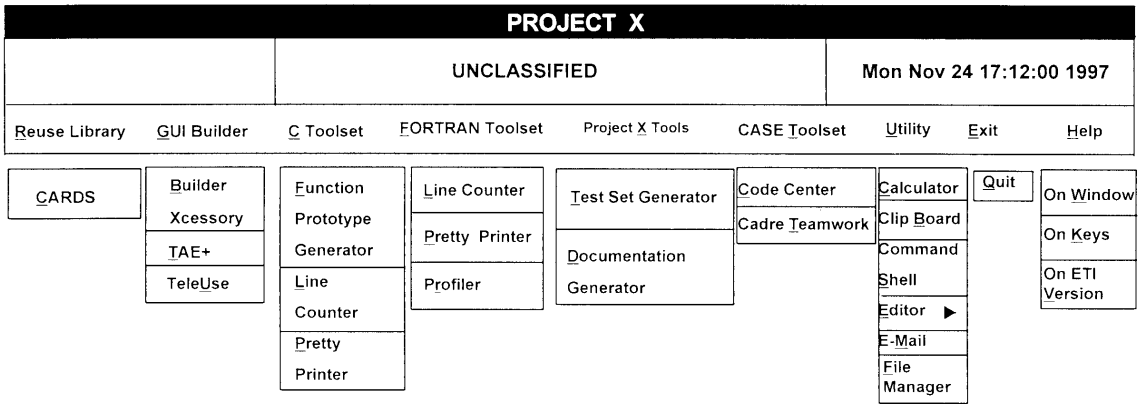

Figure A-5. Top-level window with tools and options displayed tailored to a specific project.

```
"PROJECT X"

"UNCLASSIFIED"

"Reuse Library"  "R":
       "CARDS"  "C"                              "";
"GUI Builder"  "G":
       "Builder Xcessory"  "B"                   "",
       "TAE+" "T"                                "",
       "TeleUse"  "U"                            "";
"C Toolset"  "C":
       "Function Prototype Generator"  "F" "c_fn_prototype",
       "Line Counter"  "L"                       "c_line_counter",
       "Pretty Printer"  "P"                     "c_pretty_printer";
"FORTRAN Toolset"  "F":
       "Line Counter"  "L"                       "fortran_line_counter",
       "Pretty Printer"  "P"                     "fortran_pretty_printer",
       "Profiler"  "r"                           "fortran_profiler";
"Project X Tools"  "X":
       "Test Set Generator"  "T"                 "testsetgen",
       "Documentation Generator"  "D"            "docgen";
"CASE Toolset"  "T":
       "Code Center"  "C"                        "xcodecenter",
       "Cadre Teamwork"  "T"                     "cadre";
"Utility"  "U":
       "Calculator"  "C"                         "xcalc -geometry +10+120",
       "Clip Board"  "B"                         "xclipboard geom +10+120",
       "Command Shell"  "S"                      "xterm -geom +10+120",
       "Editor" "E":
            "Ned"  "N"                           "$SWEEP_HOME/bin/ned",
            "Emacs"  "E"                         "xterm -fn 10x20 -e emacs",
            "Xedit"  "X"                         "xedit" ;  ,
       "E-mail"  "m"                             "xmh -geo +10x120",
       "File Manager"  "F"                       "file_manager";
Help:
       "On Window"  "W"                          "help.window",
       "On Keys"  "K"                            "help.keys",
       "On ETI Version"  "V"                     "help.version";
```

Figure A-6. Example of setup table tailored to a specific project.

A.4 REFERENCES

1. Naval Command, Control and Ocean Surveillance Center, RDT&E Division. 1992. "CRSS Replicated Site Procedures Manual, Alpha rel. 1.0," San Diego, CA.

2. Fernandes, K. 1992. "User Interface Specifications for Command and Control Systems," ver. 1.0, Naval Command, Control and Ocean Surveillance Center, RDT&E Division, San Diego, CA.

3. Converse, D. S., S. Gildea, S. Hardy, J. Hersh, K. Packard, R. Scheifler, and R. Swick. 1991. "User Commands for X Window System," on-line instructions, Massachusetts Institute of Technology Consortium, X ver. 11, rel. 5, Boston, MA.

APPENDIX B
USING THE TOOLS

Appendix B briefly describes the tools included with the ETI and shows how to use them. Specific examples are provided. The test files described in the examples are included with version 1.2 of the ETI. You may run the examples exactly as presented here.

B.1 C TOOLSET

B.1.1 Line Counter

Function: The line counter is the kdsi program written by B. Renaud in July 1988. It was obtained from Ohio State University. A Motif front-end was added by the ETI project. The line counter reads in text files; and then, on your monitor, displays the number of lines of code, blank lines, comment lines, and number of comments.

Example: The user provides input through the menu given in figure B-1. In this example, the user wanted the counts for all C files in the directory, `$SWEEP_HOME/testfiles.` This was indicated in the *Input* box. The user executed the tool by clicking on the *Execute* box. The center panel displays the lines of code, blank lines, comment lines, and number of comments for all C files in the designated directory.

To output counts to both your monitor and a file, you must enter a file name in the Output box.

The *Display available files* box allows you to list files.

You can find additional information on the line counter by examining the source code and readme files in the directory, `$SWEEP_HOME/tools/kdsi,` on your computer.

B.1.2 Pretty Printer

Function: This pretty printer is the C beautifier (cb) program dated September 1987 in the "SunOS Reference Manual" (in reference 1). A Motif front-end was added by the ETI project. The pretty printer reads in a C source code file and produces a source file with indentation. You can join split lines and can also split lines that are longer than a user-specified length.

Example: Name the input through the menu given in figure B-2. As shown, the user wants the file, `$SWEEP_HOME/testfiles/main.c` to be reformatted into standard C style and to be output to the file, `/mnt/sweep1.2/testfiles/reformatted_main.c.` To run the pretty printer, the user enters these file names in the appropriate boxes and clicks the option for standard C style. The output is displayed in the bottom panel of figure B-2 and is written to the designated file.

To direct the output to the screen only, omit an output file name. Options are also available to join split lines and to split lines. These options may be invoked simultaneously.

Within the ETI, you may examine the contents of the output file by going to UTILITY and Command Shell, and then typing

```
more /mnt/sweep1.2/testfiles/reformatted_main.c
```

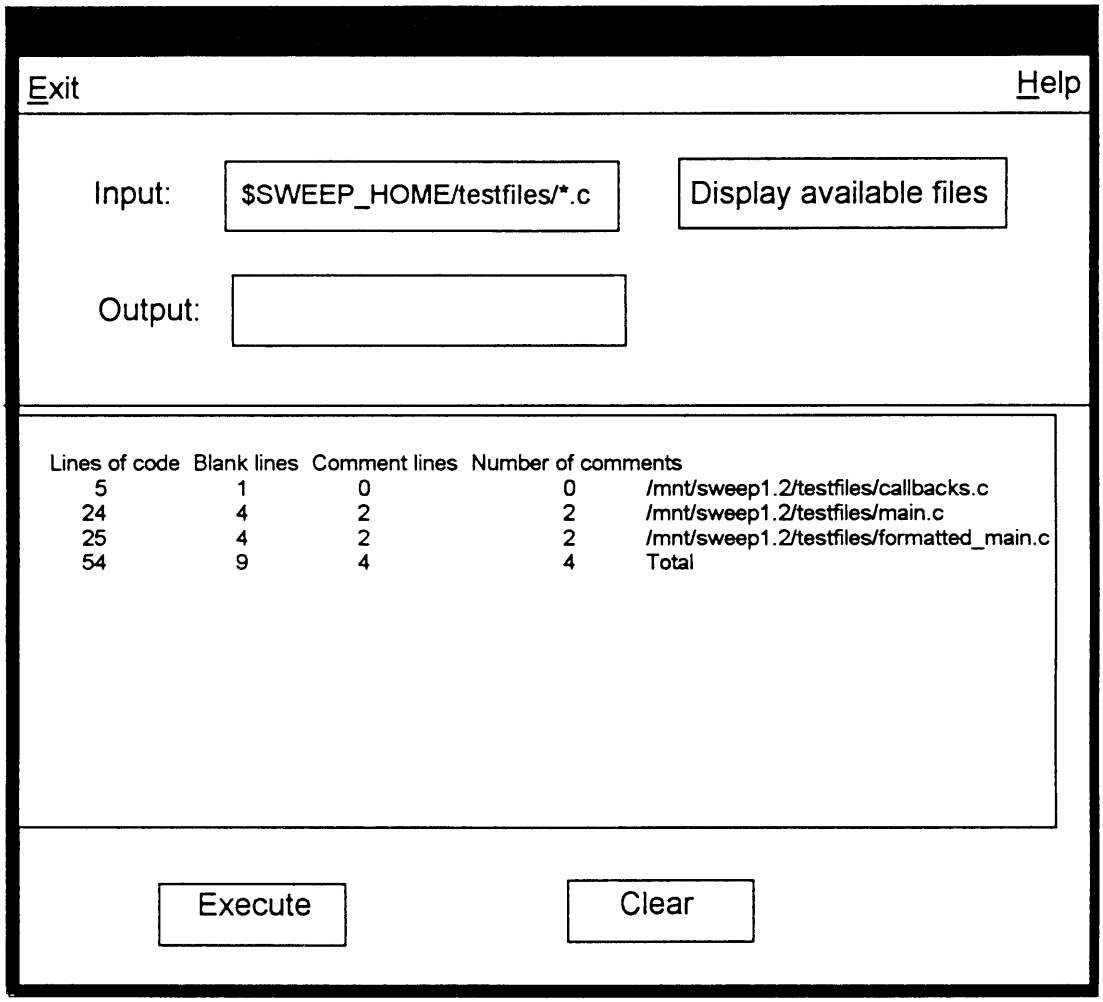

Figure B-1. Menu for the C line counter.

Additional information on the pretty printer is found in the "SunOS Reference Manual" (reference 1, Appendix B).

B.1.3 Function Prototypes Generator

Function: The function prototypes generator is the `mkproto` written by E. Smith in September 1989. It was obtained from Ohio State University. A Motif front-end was added by the ETI project. The function prototypes generator reads in one or more C source code files and produces a list of function prototypes for the functions defined in the input files.

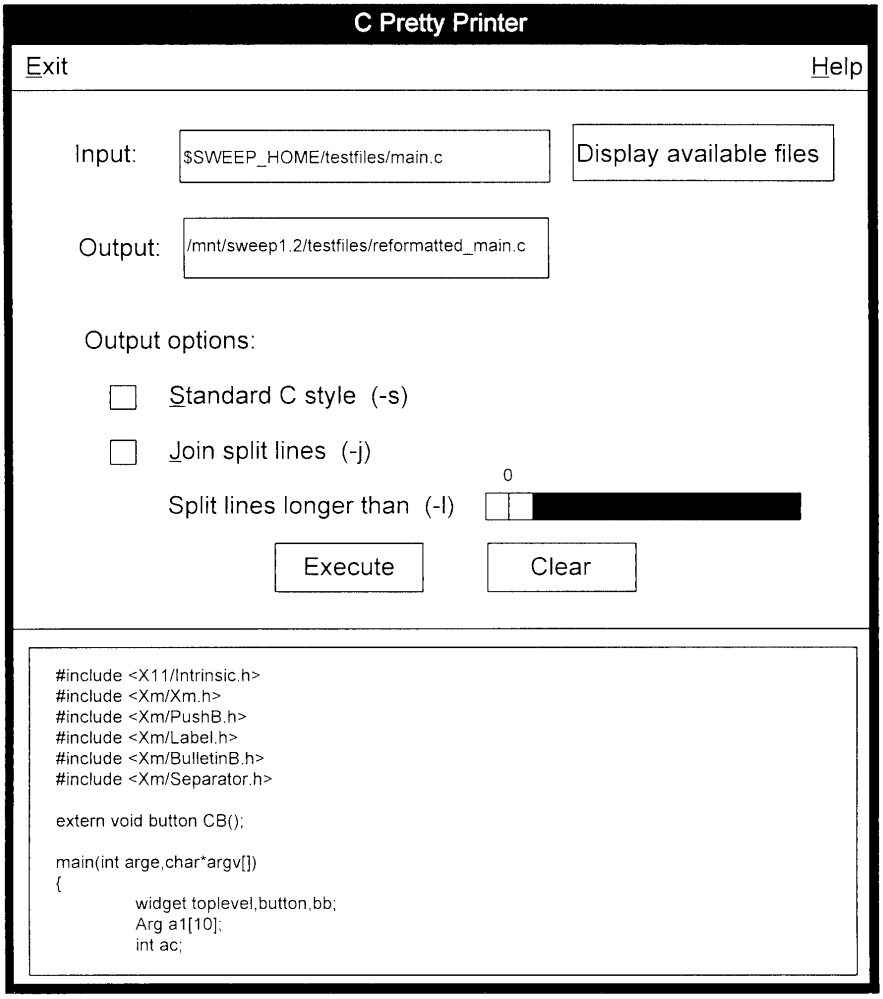

Figure B-2. Menu for C pretty printer.

Example: Figure B-3 shows how to use the function prototypes generator. Specify input files by entering the file name or names in the *Input* box or by clicking on the *Display available files* to select the files. This example generates prototypes for functions defined in the file `$SWEEP_HOME/testfiles/mkproto.c`. The *Include line number* option causes the line number (where each function was defined) to be inserted as a comment in front of the corresponding function protoype declaration. The *Execute* box was clicked on to execute the tool. The output of the function prototype declarations, with the line number inserted in front of each declaration, appears in the lower panel.

To redirect the output to a file, enter a file name in the *Output* box. Options are also available for generating prototypes for static functions and for generating code compilable by ANSI compilers only.

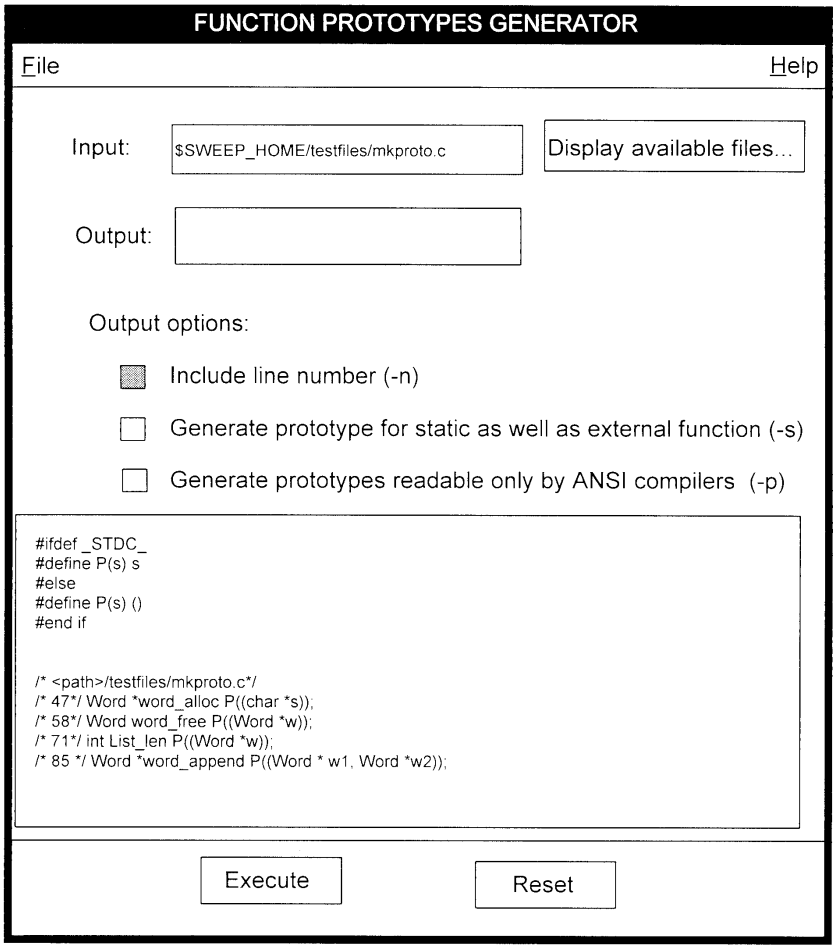

Figure B-3. Menu for the function prototypes generator.

B.2 ADA TOOLSET

B.2.1 Body Stubber

Function: The body stubber creates an Ada package body, with stubs for subprograms and tasks, from an Ada package specification containing the specifications for subprograms and tasks. The stubber reads in an Ada package specification with subprogram and task specifications and generates a file containing an Ada package body with stubs for subprograms and tasks.

Example: The following is what a typical session looks like immediately after invoking the body stubber.

```
Enter name of file: $SWEEP_HOME/testfiles/stubber_input
Enter output file
name(Default:$SWEEP_HOME/testfiles/stubber_input_body):
RETURN          — When user presses Return, the default file is created.
```

B.2.2 Compilation Order Maker

The Ada compilation order maker is a Motif front-end for invoking the Ada primitive compilation order tool (APRICOT), APRICOT report generator (ARG), and APRICOT script generator (ASG) programs. These programs are invoked from the menu shown in figure B-4.

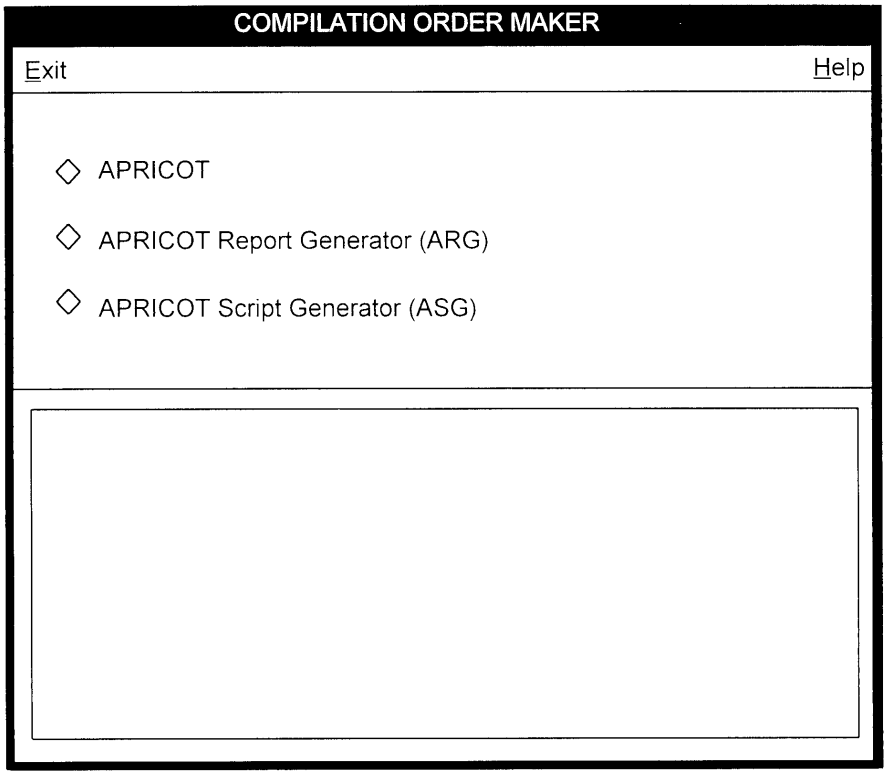

Figure B-4. Menu for the compilation order maker.

APRICOT

Function: APRICOT reads in compilable Ada source code and produces a database containing an analysis of the compilation order dependencies. The database can be used by the APRICOT report generator to generate reports based on the analysis.

Example: Figure B-5 displays the Motif front-end to APRICOT. This example splits the paged file `$SWEEP_HOME/testifies/apricot_input` into individual files and saves the analysis of the compilation order dependencies in a file for future use. To accomplish this, click on the *Select Input Files* box to select the input file; then enter the output file name, `database;` click on the *Split files* option; and then click on *Execute*.

APRICOT Report Generator (ARG)

Function: The APRICOT report generator uses a database generated by APRICOT as input and produces compilation order dependency reports.

Example: This example shows how to generate a list of file names in compilation order from the database generated by the APRICOT. Select the file, `database,` as input to ARG by clicking on the *Select Input Files* box in the menu shown in figure B-6. Then enter `include` as the output file name and select the options for generating a report containing the split-file names of all compilation program units. To execute the program, click on the *Execute* button.

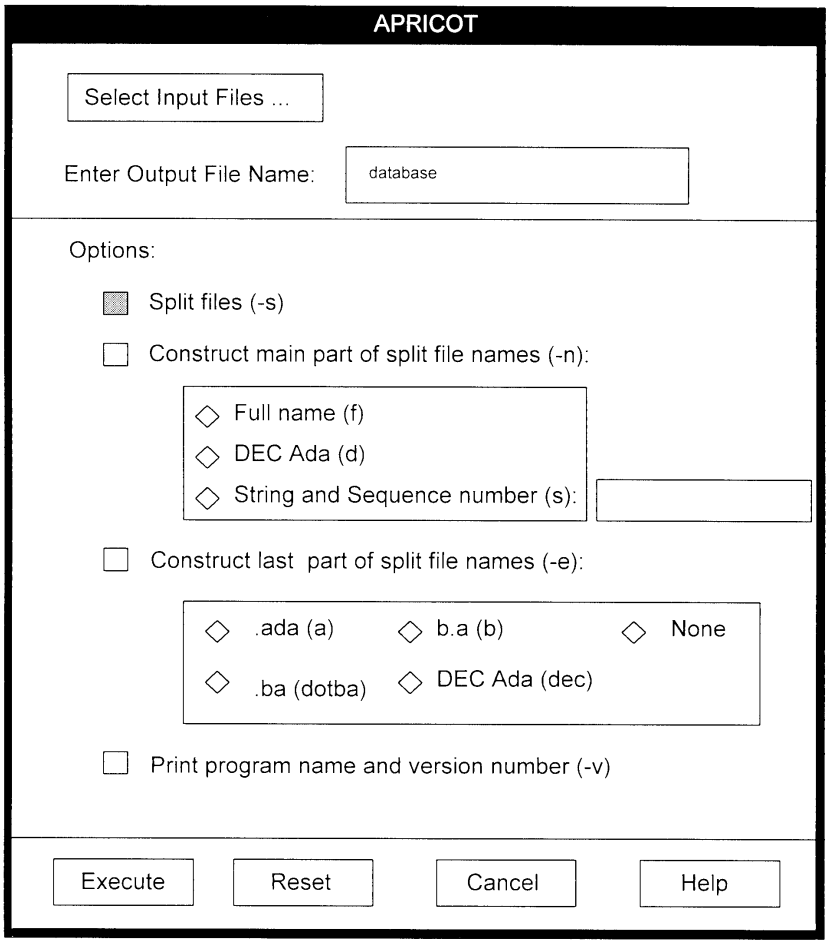

Figure B-5. Menu for APRICOT.

APRICOT Script Generator (ASG)

Function: The APRICOT script generator reads in a file containing a list of Ada file names and generates a file with a user-defined command for each file name. The output file can be a batch command file for compiling Ada source code. Compilation command files can be automatically created in this manner for any Ada compiler.

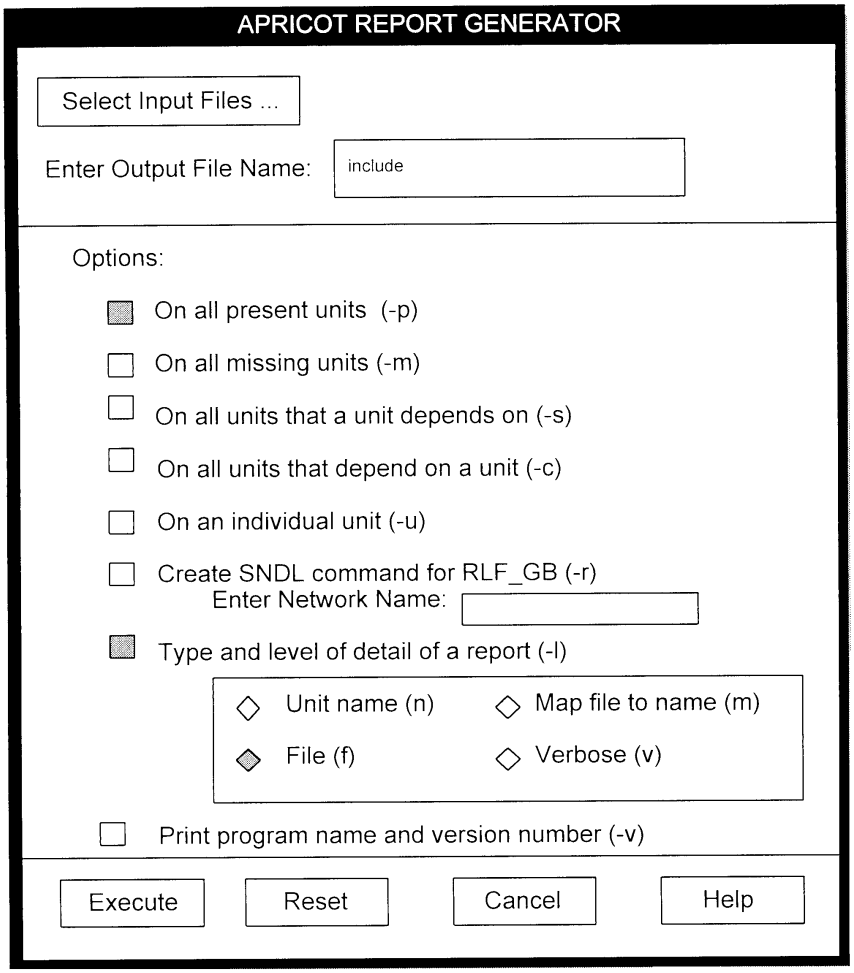

Figure B-6. Menu for the APRICOT report generator.

Example: Figure B-7 displays the ASG menu to generate a command compilation file from the `include` file produced by ARG. Click on the *Select Input Files* button to produce a file selection box where you can select the `include` file as the program input. Enter the output file, `batch,` and command, `ada -v,` in appropriate boxes. Finally, click on the *Execute* box.

B.2.3 Line Counter

Function: This line counter is a Motif front-end to the FILE_CHECKER, version 1.4. The line counter counts Ada source code statements, comments, and lines (card-image statements). Refer to Appendix C for more information on the FILE_CHECKER.

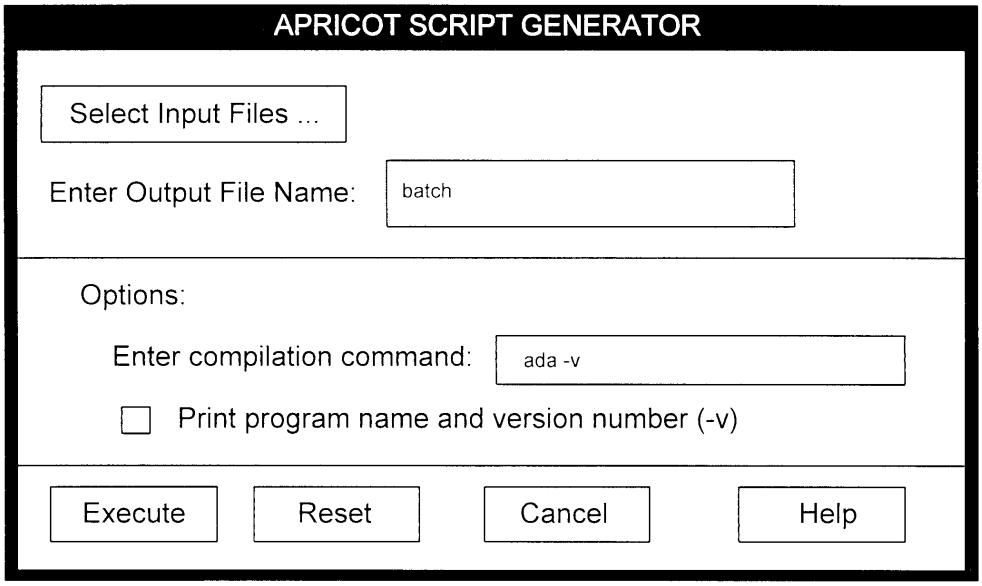

Figure B-7. Menu for the APRICOT script generator.

Example: Figure B-8 illustrates the use of the line counter. This example counts the number of lines in the Ada files in the `$SWEEP_HOME/testfiles` directory. Click on the *Select Input Files* button to select the files. The files selected are shown in the box below the button. Click on the *Execute* button to execute the program. The bottom panel displays the line counts for the selected files. Enter the file name in the *Enter Output File Name* box to save the line counts to a file.

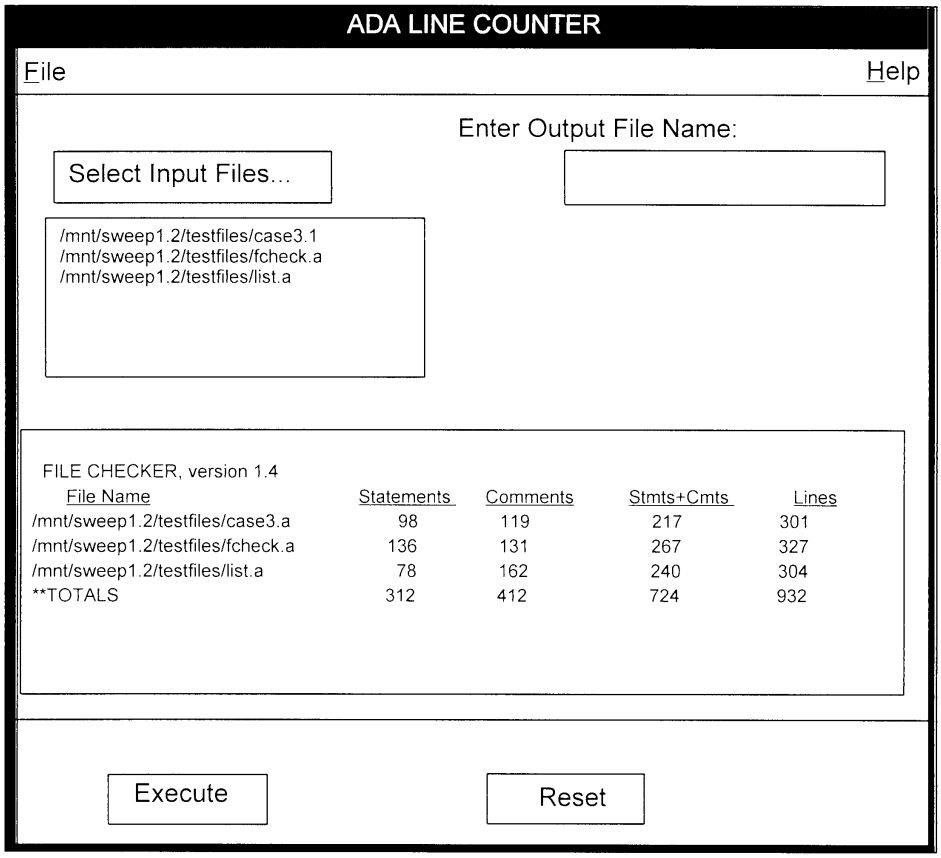

Figure B-8. Menu for the Ada line counter.

B.2.4 Pager

Function: The pager combines many files into a concatenated file and breaks a concatenated file into multiple files. It also lists all the files and their line counts from a concatenated file that contains multiple files.

Example: The example given in figure B-9 breaks a concatenated file into individual files. Select the *Unpage* option. Then click on the *Select Input Files* button to select the input file, `$SWEEP_HOME/testfiles/pager_input,` and then click on the *Execute* box. The files extracted and their line counts are displayed in the bottom panel. To redirect the output displayed on the panel to a file, enter a file in the *Enter Output File Name* box.

B.2.5 Pretty Printer

Function: The pretty printer reads in Ada source code file(s) or a file containing Ada source code file names and creates an output file with spacing, indentation, and capitalization that conforms to the proposed Ada Style Guide Handbook (reference 2, Appendix B).

Example:

The following is an example of a session after invoking the line counter.

```
Ada File? > $SWEEP_HOME/testfiles/pretty_print_input
$SWEEP_HOME/testfiles/pretty_print_inppp created
        — File of reformatted Ada source code.
```

$SWEEP_HOME/testfiles/pretty_print_inpsr created
— File containing statistical information on source code.

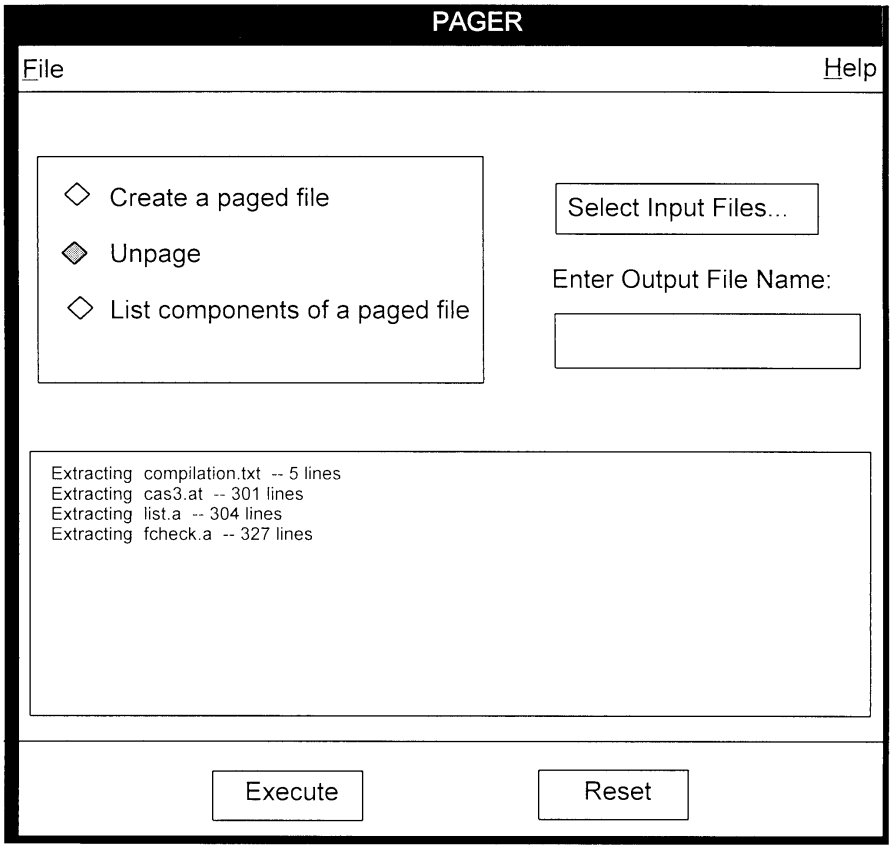

Figure B-9. Menu for pager.

B.3 UTILITY

B.3.1 Calculator, Clipboard, Command Shell, E-Mail

The calculator, clipboard, command shell, and E-mail tools are provided by the X Window System. You can find user information about these tools on line, by using `man pages`. For example, type `man xcalc, man xclipboard, man xterm, man xmh` at the Unix prompt to display manual pages for the calculator, clipboard, command shell, and e-mail, respectively.

B.3.2 Editors

B.3.2.1 NED. The Navy Command and Control System Ashore Editor (NED) is a text display and editing system that was designed for viewing portions of source code and documentation stored in the CRSS library. NED may also run as a stand-alone application for editing text files. NED was developed by K. Allen, Intermetrics, for the OSS project. NED user instructions are contained in the “CRSS Replicated Site Procedures Manual” (reference 3).

B.3.2.2 Emacs. This tool is the GNU Emacs editor from the Free Software Foundation. The editor includes facilities to send and receive mail, run subprocesses, and do compilations. It was written by R. Stallman.

B.3.2.3. Xedit. Xedit is a customizable text editor for the X Window System developed by MIT. It was written by C. Peterson, MIT X Consortium.

B.3.3 File Manager. File Manager is a Motif application used to navigate through file hierarchies. This tool provides facilities to search, move, copy, compress, and view files, and to obtain file information. It was developed by K. Allen, Intermetrics, for the OSS project.

B.4 REFERENCES

1. Sun Microsystems, Inc. 1990. “SunOS Reference Manual, rev. A.”
2. “Ada Style Guide Handbook.” 1988. MIL-HDBK-1804, Draft, National Aeronautics and Space Administration/Goddard Space Flight Center.
3. Naval Command, Control and Ocean Surveillance Center, RDT&E Division. 1992. “CRSS Replicated Site Procedures Manual, Alpha rel. 1.0,” San Diego, CA.

APPENDIX C
BACKGROUND INFORMATION ON THE ADA TOOLS

APRICOT, APRICOT Report Generator (ARG), and APRICOT Script Generator (ASG)

APRICOT and ARG are tools for determining the compilation order for a collection of Ada files. They also perform pager functions. ASG creates command compilation files from concatenated pager files. These tools were developed by R. Ollerton in 1992, and updated by him in 1993. They currently run on Vax/VMS and SPARCstations.

Body Stubber

The body stubber is Body Stubber 2 from the AdaNet repository, Morgan town, WV. It was written by J. Orost, Concurrent Computer Corporation in 1983, and updated by him in 1987. N. Tran, NRaD, upgraded it to run on VAX/VMS (DEC Ada), Sun (Verdix), and other validated Ada compilers. This tool is especially useful for large projects, where interfaces must be defined very early. The body stubber reads in an Ada package specification containing the specification for subprograms and tasks. It automatically creates a compilable package body containing stubs for the subprograms and tasks. NRaD resubmitted it to the AdaNet Repository in December 1991. It was submitted again to AdaNet in December 1993, because it was misplaced at AdaNet during a move.

Line Counter

The Ada line counter is `FILE_CHECKER,` version 1.4, from the AdaNet repository. It was written by R. Conn, Texas Instruments (TI) and Management Assistance Corporation of America (MACA) in 1985 and updated in 1989. Fixes have been made by H. Mumm, NRaD, and P. Babick, Science Applications International Corporation (SAIC). The line counter is written in Ada and runs on VAX/VMS (DEC Ada), Sun (Verdix), and other validated Ada compilers. The tool counts Ada source lines of code several ways. This tool was used on the Ada Bit-Oriented Message Handler (ABOM) project by NRaD and SAIC to report programmer productivity statistics. It was submitted to the AdaNet in December 1991 and was also again submitted to AdaNet in December 1993, because it was misplaced.

Pager

The pager includes the Pager2 program, which was developed in 1989 by R. Conn, of TI and MACA, and a Motif front-end written by N. Tran, NRaD. Pager2 was written in Ada and runs on VAX/VMS (DEC Ada), Verdix (Sun), IntegrAda, and other validated Ada compilers. The tool concatenates Ada source files, breaks concatenated files into individual files, and makes listings. Pager requires that source files be in the pager format, a prefixed banner of comment statements. Pager is a tool used for storing and transporting related files. It is used by AdaNet users to break apart concatenated files.

Pretty Printer

This pretty printer is pretty printer 6 from the AdaNet repository. It was written in 1987 and updated in 1988 by A. Shell, AdaCraft, Incorporated, for the NASA/Goddard Space Flight

Center. It was written in Ada and runs on VAX/VMS (DEC Ada), SUN (Verdix), PC (Alsys), and on other compilers/computers. Pretty printer 6 implements many of the directives in the proposed MIL-HDBK-1804, “Ada Style Guide.” The modifications required to change the number of columns of indentation, the case of variable names, and other pretty printer parameters are isolated in one Ada package specification. This pretty printer was used by NRaD for the Ada discrete-event simulation research conducted by the Shared Adaptive Internetworking (SAINT) project.

APPENDIX D

SOURCES FOR PUBLIC DOMAIN TOOLS—PHONE NUMBERS AND ADDRESSES

Repository	Telephone Number to Request an Account	Address
STARS ASSET Library	(304) 594–1762	STARS ASSET Library 2611 Cranberry Square Morgantown, WV 26505
AdaNET	(800) 444–1458	MountainNet, Inc 2705 Cranberry Square Morgantown, WV 26505
Archie	None required	archie.ans.net (147.225.1.10)